

Singleton Pattern

Friday, March 12, 2021 6:11 PM

Software engineering concept where one allows global access to a class and that class only exists once.

Ex. Manager Classes: Game Manager, Player Manager, Item Manager, UI Manager, etc.

Instead of using GetComponent the class can be accessed directly.

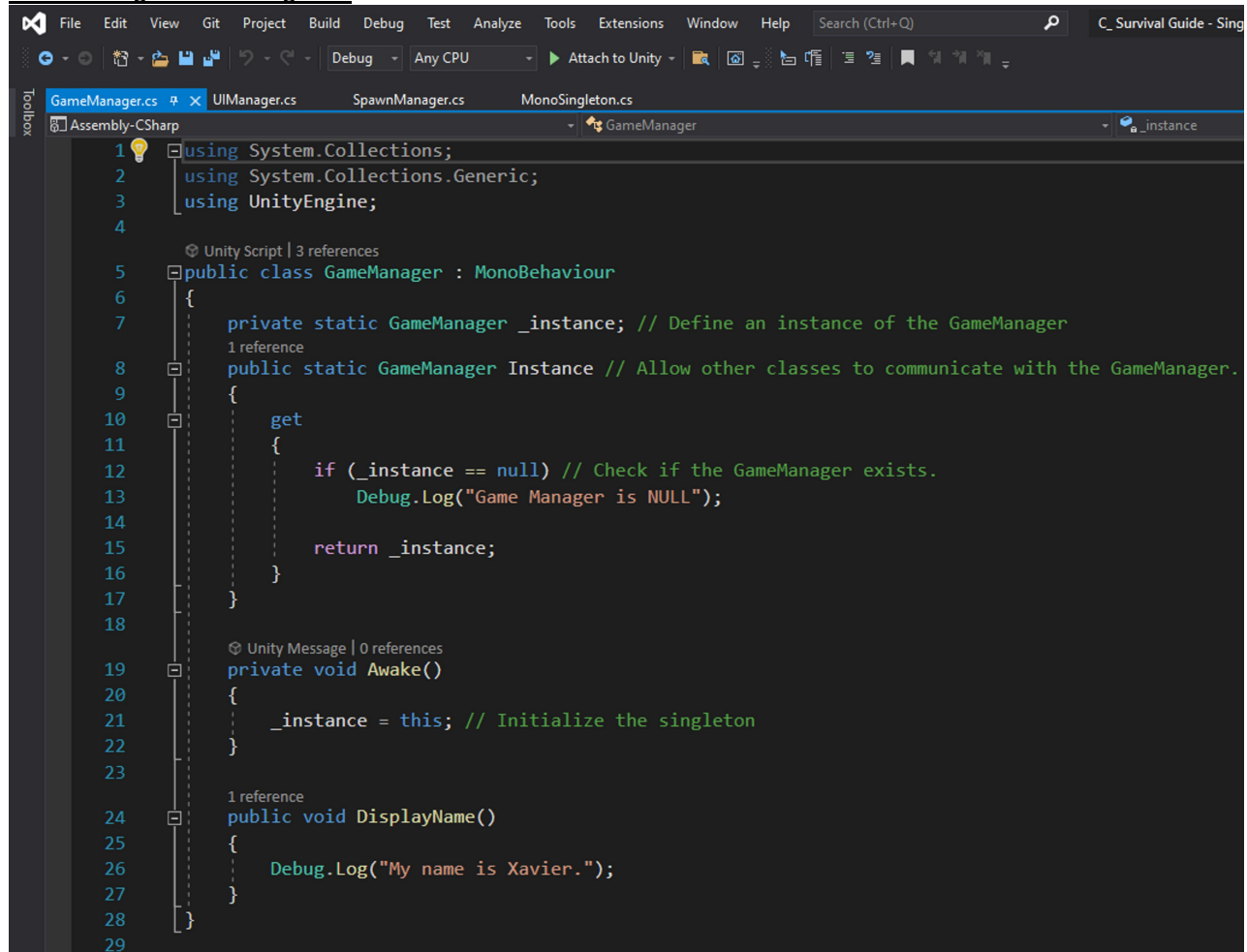
(If the game has only one spawn manager then why not make it a singleton?)

Manager classes should be communicated with, not the other way around!

Two Major Benefits of Singletons are:

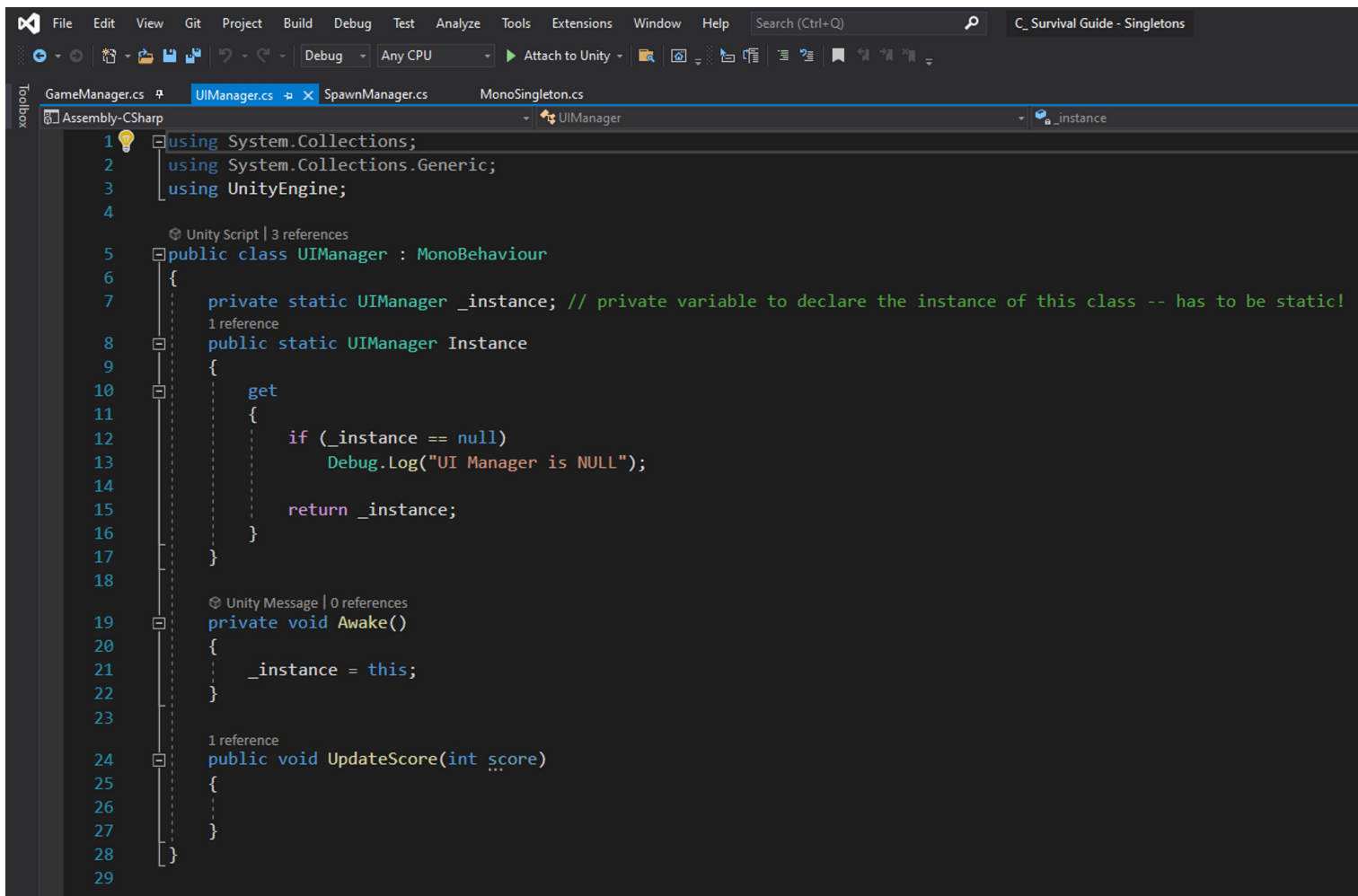
- Global access communication
- Lazy instantiation

Game Manager & UI Manager Ex.

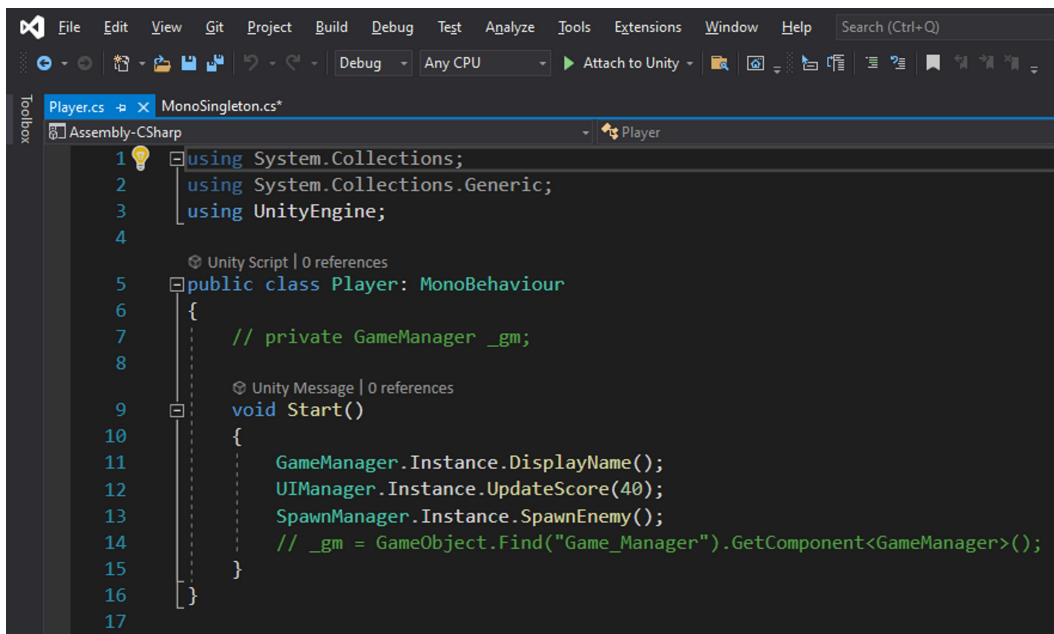


The screenshot shows the Visual Studio IDE with the 'GameManager.cs' file open. The code implements a Singleton pattern for a 'GameManager' class. It includes using statements for 'System.Collections', 'System.Collections.Generic', and 'UnityEngine'. The class 'GameManager' inherits from 'MonoBehaviour'. It features a private static '_instance' variable, a public static 'Instance' property with a getter that checks for null and logs a message if it is, and an 'Awake()' method that initializes the instance to 'this'. There is also a 'DisplayName()' method that logs the name 'Xavier'.

```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 public class GameManager : MonoBehaviour
6 {
7     private static GameManager _instance; // Define an instance of the GameManager
8     public static GameManager Instance // Allow other classes to communicate with the GameManager.
9     {
10         get
11         {
12             if (_instance == null) // Check if the GameManager exists.
13                 Debug.Log("Game Manager is NULL");
14
15             return _instance;
16         }
17     }
18
19     private void Awake()
20     {
21         _instance = this; // Initialize the singleton
22     }
23
24     public void DisplayName()
25     {
26         Debug.Log("My name is Xavier.");
27     }
28 }
29
```



```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 public class UIManager : MonoBehaviour
6 {
7     private static UIManager _instance; // private variable to declare the instance of this class -- has to be static!
8     public static UIManager Instance
9     {
10         get
11         {
12             if (_instance == null)
13                 Debug.Log("UI Manager is NULL");
14
15             return _instance;
16         }
17     }
18
19     private void Awake()
20     {
21         _instance = this;
22     }
23
24     public void UpdateScore(int score)
25     {
26     }
27 }
28
29
```



```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 public class Player : MonoBehaviour
6 {
7     // private GameManager _gm;
8
9     void Start()
10     {
11         GameManager.Instance.DisplayName();
12         UIManager.Instance.UpdateScore(40);
13         SpawnManager.Instance.SpawnEnemy();
14         // _gm = GameObject.Find("Game_Manager").GetComponent<GameManager>();
15     }
16 }
17
```